

Robust deep networks with randomized tensor regression layers

Arinbjörn Kolbeinsson *
Samsung AI, Cambridge
Imperial College London

Jean Kossaifi *
Samsung AI, Cambridge
Imperial College London

Yannis Panagakis
Samsung AI, Cambridge
Imperial College London

Adrian Bulat
Samsung AI, Cambridge

Anima Anandkumar
California Institute of Technology

Ioanna Tzoulaki
Imperial College London

Paul Matthews
Imperial College London

Abstract

In this paper, we propose a novel randomized tensor decomposition for tensor regression. It allows to stochastically approximate the weights of tensor regression layers by randomly sampling in the low-rank subspace. We theoretically and empirically establish the link between our proposed stochastic rank-regularization and the dropout on low-rank tensor regression. This acts as an additional stochastic regularization on the regression weight, which, combined with the deterministic regularization imposed by the low-rank constraint, improves both the performance and robustness of neural networks augmented with it. In particular, it makes the model more robust to adversarial attacks and random noise, without requiring any adversarial training. We perform thorough study of our method on synthetic data, object classification on the CIFAR100 and ImageNet datasets, and large scale brain-age prediction on UK Biobank brain MRI dataset. We demonstrate superior performance in all cases, as well as significant improvement in robustness to adversarial attacks and random noise.

1. Introduction

Deep neural networks have been evolved to powerful predictive models with remarkable performance on computer vision tasks [26, 30, 16]. Such models are usually over-parameterized, involving an enormous number (possibly millions) of parameters. This is much larger than the typical number of available training samples, making deep networks prone to overfitting [5]. Coupled with overfitting, deep networks lack robustness to small adversarial or noise perturbations [14]. In computer vision tasks, such as image classification or regression, perturbed examples are often

perceived identically to the original ones by humans while lead arbitrarily different predictions by networks. These shortcomings pose a possible obstacle for mass deployment of systems relying on deep learning in sensitive fields such as medical image analysis for disease prediction and expose an inherent weakness in their reliability.

To improve the robustness of deep neural networks to (adversarial or random) perturbations and prevent them from overfitting, several methods that essentially affect the parameters of deep networks via regularization have been investigated. One line of research includes methods that apply regularization functions (e.g., ℓ_2 - or ℓ_1 - norm) on networks parameters [32, 27, 36, 49]. Besides the aforementioned general-purpose regularization functions, neural networks specific methods such as early stopping of back-propagation [5], batch normalization [17], dropout [38] and its variants –e.g., DropConnect [44]– are algorithmic approaches to reducing overfitting in over-parametrized networks and have been widely adopted in practice. An alternative approach for reducing the number of effective parameters in deep nets relies on low-rank regularization. That is, by leveraging the redundancy in network parameters, methods such as [41, 7, 47, 23] employ low-rank approximations of deep networks weight matrices (or tensors) and hence massively reduce the number of learnable parameters. Links between regularization and robustness of deep neural networks to adversarial perturbations have been recently established in [2, 18].

In this paper, we introduce a novel low-rank inducing regularization method for improving robustness of deep networks to adversarial and random perturbations. We replace flattening and fully-connected layers entirely with a randomized tensor regression layer (R-TRL). This preserves the structure by expressing an output tensor as the result of a tensor contraction between the input tensor and some low-rank regression weight tensors. We propose to obtain

*Equal contribution

the low-rank regression weights by solving a novel randomized tensor regression model, which leads to the stochastic reduction of the rank, either by a fixed percentage during training or according to a series of Bernoulli random variables. This is akin to dropout, which, by randomly dropping units during training, prevents over-fitting. However, rather than dropping random elements from the *activation* tensor, this is done on rank of the regression weight tensor. We conduct thorough experiments in image classification and phenotypic trait prediction from MRI analysis. Specifically, we apply our method to the UK Biobank brain MRI dataset for estimating brain-age, which has been associated with a range of diseases and mortality [9], and could be an early predictor for Alzheimer’s disease [13]. A more accurate and more robust brain age estimation can consequently lead to more accurate disease diagnoses.

In summary, we make the following contributions:

- We propose a novel randomized TRL based on a stochastic tensor decomposition
- We explore two schemes: (i) selecting random element to keep in the low-rank subspace, following a Bernoulli distribution and (ii) keeping a random subset of the *fibers* of the tensor, with replacement.
- We theoretically and empirically establish the link between the proposed randomized TRL and dropout on the deterministic low-rank tensor regression.
- We demonstrate state-of-the-art results for image classification on CIFAR100 and ImageNet, and show superior performance to both the regular network with fully-connected layer and with tensor regression layer without our proposed decomposition.
- Our method makes neural networks significantly more robust to adversarial noise, *without* adversarial training.
- We also demonstrate a large performance improvement (more than 20%) on the regression task using MRI data, with 3D-ResNet and our proposed R-TRL, compared to a regular 3D-ResNet. Our R-TRL is also significantly more robust to random noise in the input.

2. Closely related work

Network regularization and dropout. Several methods that improve generalization by mitigating overfitting have been developed in the context of deep learning. The interested reader is referred to the work of [28] and the references therein for a comprehensive survey of regularization techniques for deep networks. The most closely related regularization method to our approach is Dropout [38], which is probably the most widely adopted technique for training neural networks while preventing overfitting. Concretely, during dropout training each unit (i.e., neuron) is equipped with a binary Bernoulli random variable and only the networks weights whose corresponding Bernoulli variables are sampled with value 1 are updated at each back-

propagation step. At each iteration, those Bernoulli variables are re-sampled again and the weights are updated accordingly. The proposed regularization method can be interpreted as dropout on low-rank tensor regression, a fact which we proved in Section 4.2.

Randomized tensor decompositions. Tensor decompositions exhibit high computational cost and low convergence rate when applied to massive multi-dimensional data. To accelerate computation, and enable them to scale, randomized approaches have been employed. A randomized least squares algorithm for CP decomposition is proposed by [1], which is significantly faster than traditional CP decomposition. In [12], CP is applied on a small tensor generated by multi-linear random projection of the high-dimensional tensor. The CP decomposition of the large-scale tensor is obtained by back projection of the CP decomposition of the small tensor. [46] introduce a fast yet provable randomized CP decomposition that performs randomized tensor contraction using FFT. Methods in [37, 43] are highly computationally efficient algorithms for computing large-scale CP decompositions by applying random projections into a set of small tensors, derived by subdividing a tensor into a set of blocks. Fast randomized algorithms that employ sketching for approximating Tucker decomposition have been also investigated [42, 50]. More recently, a randomized tensor ring decomposition that employs tensor random projections has been developed in [48]. The most similar method to ours is that of [1], where elements of the tensor are sampled randomly, and each factor of the decomposition updated in an iterative manner. By contrast, our method allows for end-to-end training, and applies randomization on the *fibers* of the tensor, effectively randomizing the rank of the weight tensor.

Tensor Regression Layers. Reducing the storage and computational costs of deep networks has become critical for meeting the requirements of environments with limited memory or computational resources. To this end, a surge of network compression and approximation algorithms have recently been proposed in the context of deep learning. By leveraging the redundancy in network parameters, methods such as [41, 7, 47, 23] employ low-rank approximations of deep networks weight matrices (or tensors) for parameter reduction. Network compression methods in the frequency domain [6] have also been investigated. An alternative approach for reducing the number of effective parameters in deep nets relies on sketching, whereby, given a matrix or tensor of input data or parameters, one first compresses it to a much smaller matrix (or tensor) by multiplying it by a (usually) random matrix with certain properties [20, 10].

A particularly appealing approach to network compression, especially for visual data (and other types of multidimensional and multi-aspect data), is tensor regression networks [23]. Deep neural networks typically leverage

the spatial structure of input data via series of convolutions, point-wise non-linearities, pooling, etc. However, this structure is usually wasted by the addition, at the end of the networks' architectures, of a flattening layer followed by one or several fully-connected layers. A recent line of study focuses on alleviating this using tensor methods. [22] proposed tensor contraction as a layer, to reduce the size of activation tensors, and demonstrated large space savings by replacing fully-connected layers with this layer. However, a flattening layer and fully-connected layers were still ultimately needed for producing the outputs. Recently, tensor regression networks [23] propose to replace these entirely with a tensor regression layer (TRL). This preserves the structure by expressing an output tensor as the result of a tensor contraction between the input tensor and some low-rank regression weight tensors. In addition, these allow for large space savings without sacrificing accuracy. [4] explore the same model with various low-rank structures on the regression weight tensor.

3. Multilinear algebra background

In this section, we introduce the notations and notions necessary to introduce our stochastic rank regularization.

Notation: We denote \mathbf{v} vectors (1st order tensors) and \mathbf{M} matrices (2nd order tensors). \mathbf{Id} is the identity matrix. We denote \mathcal{X} tensors of order $N \geq 3$, and denote its element (i, j, k) as $\mathcal{X}_{i_0, i_1, \dots, i_{N-1}}$ or $\mathcal{X}(i_0, i_1, \dots, i_{N-1})$. A colon is used to denote all elements of a mode e.g. the mode-0 fibers of \mathcal{X} are denoted as $\mathcal{X}_{:, i_1, \dots, i_{N-1}}$. The transpose of \mathbf{M} is denoted \mathbf{M}^\top . Finally, for any $i, j \in \mathbb{N}, i < j, [i \dots j]$ denotes the set of integers $\{i, i+1, \dots, j-1, j\}$, and $i \text{ div } j$ the integer division of i by j .

Tensor unfolding: Given a tensor, $\mathcal{X} \in \mathbb{R}^{I_0 \times I_1 \times \dots \times I_{N-1}}$, its mode- n unfolding is a matrix $\mathbf{X}_{[n]} \in \mathbb{R}^{I_n \times M}$, with $M = \prod_{k=0, k \neq n}^{N-1} I_k$ and is defined by the mapping from element (i_0, i_1, \dots, i_N) to (i_n, j) , with $j = \sum_{k=0, k \neq n}^{N-1} i_k \times \prod_{m=k+1, m \neq n}^{N-1} I_m$.

Tensor vectorization: Given a tensor, $\mathcal{X} \in \mathbb{R}^{I_0 \times I_1 \times \dots \times I_{N-1}}$, we can flatten it into a vector $\text{vec}(\mathcal{X})$ of size $(I_0 \times \dots \times I_{N-1})$ defined by the mapping from element $(i_0, i_1, \dots, i_{N-1})$ of \mathcal{X} to element j of $\text{vec}(\mathcal{X})$, with $j = \sum_{k=0}^{N-1} i_k \times \prod_{m=k+1}^{N-1} I_m$.

Mode- n product: For a tensor $\mathcal{X} \in \mathbb{R}^{I_0 \times I_1 \times \dots \times I_{N-1}}$ and a matrix $\mathbf{M} \in \mathbb{R}^{J \times I_n}$, the n -mode product of a tensor is a tensor of size $(I_0 \times \dots \times I_{n-1} \times J \times I_{n+1} \times \dots \times I_{N-1})$ and can be expressed using unfolding of \mathcal{X} and the classical dot product as: $(\mathcal{X} \times_n \mathbf{M})_{[n]} = \mathbf{M} \mathbf{X}_{[n]} \in \mathbb{R}^{I_0 \times \dots \times I_{n-1} \times R \times I_{n+1} \times \dots \times I_{N-1}}$

Generalized inner product: For two tensors $\mathcal{X} \in \mathbb{R}^{K_0 \times \dots \times K_x \times I_0 \times \dots \times I_{N-1}}$ and $\mathcal{Y} \in \mathbb{R}^{I_0 \times \dots \times I_{N-1} \times L_0 \times \dots \times L_y}$, we denote by

$\langle \mathcal{X}, \mathcal{Y} \rangle_N \in \mathbb{R}^{K_0 \times \dots \times K_x \times I_{N-1} \times L_0 \times \dots \times L_y}$ the contraction of \mathcal{X} by \mathcal{Y} along their $N-1$ last (respectively first) modes. $\langle \mathcal{X}, \mathcal{Y} \rangle_N = \sum_{i_0=0}^{I_0} \sum_{i_1=0}^{I_1} \dots \sum_{i_{N-1}=0}^{I_{N-1}} \mathcal{X}_{\dots, i_0, i_1, \dots, i_{N-1}} \mathcal{Y}_{i_0, i_1, \dots, i_{N-1}, \dots}$

Kruskal tensor: Given a tensor $\mathcal{X} \in \mathbb{R}^{I_0 \times I_1 \times \dots \times I_{N-1}}$, the Canonical-Polyadic decomposition (CP), also called PARAFAC, decomposes it into a sum of R rank-1 tensors. The number of terms in the sum, R , is known as the rank of the decomposition. Formally, we find the vectors $\mathbf{u}_k^{(0)}, \mathbf{u}_k^{(1)}, \dots, \mathbf{u}_k^{(N-1)}$, for $k = [0 \dots R-1]$, as well as an optional vector of weights $\boldsymbol{\lambda} \in \mathbb{R}^R$ to absorb the magnitude of the factors such that:

$$\mathcal{X} = \sum_{k=0}^{R-1} \underbrace{\lambda_k \mathbf{u}_k^{(0)} \circ \mathbf{u}_k^{(1)} \circ \dots \circ \mathbf{u}_k^{(N-1)}}_{\text{rank-1 components}}, \quad (1)$$

For any $k \in [0 \dots N-1]$, these vectors $\mathbf{u}_r^{(k)}, r \in [0 \dots R-1]$ can be collected in matrices, called factors or the decomposition, and defined as $\mathbf{U}^{(k)} = [\mathbf{u}_0^{(k)}, \mathbf{u}_1^{(k)}, \dots, \mathbf{u}_{R-1}^{(k)}]$.

The decomposition can be denoted more compactly as $\mathcal{X} = \llbracket \mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N-1)} \rrbracket$, or $\mathcal{X} = \llbracket \boldsymbol{\lambda}; \mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N-1)} \rrbracket$ if a weights vector is used.

Tucker tensor: Given a tensor $\mathcal{X} \in \mathbb{R}^{I_0 \times I_1 \times \dots \times I_{N-1}}$, we can decompose it into a low rank core $\mathcal{G} \in \mathbb{R}^{R_0 \times R_1 \times \dots \times R_{N-1}}$ by projecting along each of its modes with projection factors $(\mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N-1)})$, with $\mathbf{U}^{(k)} \in \mathbb{R}^{R_k \times I_k}, k \in (0, \dots, N-1)$. The tensor in its decomposed form can then be written:

$$\begin{aligned} \mathcal{X} &= \mathcal{G} \times_0 \mathbf{U}^{(0)} \times_1 \mathbf{U}^{(2)} \times \dots \times_{N-1} \mathbf{U}^{(N-1)} \\ &= \llbracket \mathcal{G}; \mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N-1)} \rrbracket \end{aligned} \quad (2)$$

Note that the Kruskal form of a tensor can be seen as a Tucker tensor with a super-diagonal core.

Tensor diagrams: In order to represent easily tensor operations, we adopt the tensor diagrams, where tensors are represented by vertices (circles) and edges represent their modes. The degree of a vertex then represents its order. Connecting two edges symbolizes a tensor contraction between the two represented modes. Figure 1 presents a tensor diagram of the tensor regression layer and its stochastic rank-regularized counter-part.

4. Randomized tensor regression layer

In this section, we introduce the randomized tensor regression layer. Specifically, we propose a new stochastic rank-regularization, applied to low-rank tensors in decomposed forms. This formulation is general and can be applied to any type of decomposition. We introduce it here, without loss of generality, to the case of Tucker and CP decompositions.

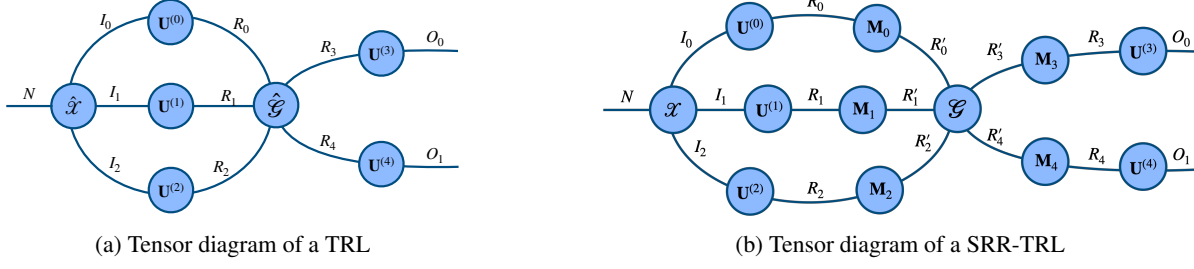


Figure 1: Tensor diagrams of the TRL (left) and our proposed SRR-TRL (right), with low-rank constraints imposed on the regression weights tensor using a Tucker decomposition. Note that the CP case is readily given by this formulation by additionally having the core tensor \mathcal{G} be super-diagonal, and setting $\mathbf{M} = \mathbf{M}^{(0)} = \dots = \mathbf{M}^{(N)} = \text{diag}(\lambda)$.

Tensor regression layer Let us denote by $\mathcal{X} \in \mathbb{R}^{I_0 \times I_1 \times \dots \times I_{N-1}}$ the input activation tensor for a sample and $\mathbf{y} \in \mathbb{R}^{I_N}$ the label vector. A tensor regression layer estimates the regression weight tensor $\mathcal{W} \in \mathbb{R}^{I_0 \times I_1 \times \dots \times I_N}$, expressed under some low-rank decomposition.

For instance, if a Tucker decomposition with rank (R_0, \dots, R_N) is used, we have:

$$\mathbf{y} = \langle \mathcal{X}, \mathcal{W} \rangle_N + \mathbf{b} \quad (3)$$

with $\mathcal{W} = \mathcal{G} \times_0 \mathbf{U}^{(0)} \times_1 \mathbf{U}^{(1)} \dots \times_N \mathbf{U}^{(N)}$

with $\mathcal{G} \in \mathbb{R}^{R_0 \times \dots \times R_N}$, $\mathbf{U}^{(k)} \in \mathbb{R}^{I_k \times R_k}$ for each k in $[0 \dots N]$ and $\mathbf{U}^{(N)} \in \mathbb{R}^{O \times R_N}$.

A new randomized tensor decomposition We propose a novel randomized decomposition on \mathcal{W} : for any $k \in [0 \dots N]$, let $\mathbf{M}^{(k)} \in \mathbb{R}^{R_0 \times R_0}$ be a sketch matrix (e.g. a random projection or column selection matrix) and, $\tilde{\mathbf{U}}^{(k)} = \mathbf{U}^{(k)}(\mathbf{M}^{(k)})^\top$ be a sketch of factor matrix $\mathbf{U}^{(k)}$, and $\tilde{\mathcal{G}} = \mathcal{G} \times_0 \mathbf{M}^{(0)} \times \dots \times_N \mathbf{M}^{(N)}$ a sketch of the core tensor \mathcal{G} .

We can now express our Randomized-Tensor Regression Layer (R-TRL). Given an activation tensor $\mathcal{X} \in \mathbb{R}^{I_0 \times \dots \times I_{N-1}}$ and a target label vector $\mathbf{y} \in \mathbb{R}^{I_N}$, a R-TRL layer is written from equation 3 as follows:

$$\mathbf{y} = \langle \mathcal{X}, \tilde{\mathcal{W}} \rangle_{N-1} \quad (4)$$

with $\tilde{\mathcal{W}}$ being a stochastic approximation of Tucker decomposition, namely:

$$\tilde{\mathcal{W}} = \tilde{\mathcal{G}} \times_0 \tilde{\mathbf{U}}^{(0)} \times \dots \times_N \tilde{\mathbf{U}}^{(N)} \quad (5)$$

Even though several sketching methods have been proposed, we focus here on SRR with two different types of binary sketching matrices, namely binary matrix sketching with replacement and binary diagonal matrix sketching with Bernoulli entries, which we detail below.

4.1. R-TRL with replacement:

In this setting, we introduce the SRR with binary sketching matrix (with replacement). We first choose $\theta \in [0, 1]$.

Mathematically, we introduce the uniform sampling matrices $\mathbf{M}^{(0)} \in \mathbb{R}^{R_0 \times R_0}, \dots, \mathbf{M}^{(N)} \in \mathbb{R}^{R_N \times R_N}$. \mathbf{M}_j is a uniform sampling matrix, selecting K_j elements, where $K_j = R_j \text{div } \theta$. In other words, for any $i \in [0 \dots N]$, $\mathbf{M}^{(i)}$ verifies:

$$\mathbf{M}^{(i)}(j, :) = \begin{cases} \mathbf{0} & \text{if } j > K \\ \text{Id}_m(r, :), m \in [0 \dots R_i] & \text{otherwise} \end{cases} \quad (6)$$

Note that in practice this product is never explicitly computed, we simply select the correct elements from \mathcal{G} and its corresponding factors.

4.2. R-TRL with Bernoulli sampling

In this setting, we introduce the SRR with diagonal binary sketching matrix with Bernoulli entries.

Bernoulli Tucker randomized tensor regression For any $n \in [0 \dots N]$, let $\lambda^{(n)} \in \mathbb{R}_n^R$ be a random vector, the entries of which are i.i.d. Bernoulli(θ), then a diagonal Bernoulli sketching matrix is defined as $\mathbf{M}^{(n)} = \text{diag}(\lambda^{(n)})$.

When the low-rank structure on the weight tensor \mathcal{W} of the TRL is imposed using a Tucker decomposition, the randomized Tucker approximation is expressed as:

$$\begin{aligned} \tilde{\mathcal{W}} &= \mathcal{G} \times_0 \mathbf{M}^{(0)} \times \dots \times_{N+1} \mathbf{M}^{(N)} \\ &\times_0 \left(\mathbf{U}^{(0)}(\mathbf{M}^{(0)})^\top \right) \times \dots \times_{N+1} \left(\mathbf{U}^{(N)}(\mathbf{M}^{(N)})^\top \right) \\ &= \llbracket \tilde{\mathcal{G}}; \tilde{\mathbf{U}}^{(0)}, \dots, \tilde{\mathbf{U}}^{(N)} \rrbracket \end{aligned} \quad (7)$$

The main advantage of considering the above-mentioned sampling matrices is that the products $\tilde{\mathbf{U}}^{(k)} = \mathbf{U}^{(k)}(\mathbf{M}^{(k)})^\top$ or $\tilde{\mathcal{G}} = \mathcal{G} \times_0 \mathbf{M}^{(0)} \times \dots \times_N \mathbf{M}^{(N)}$ are never explicitly computed, we simply select the elements from \mathcal{G} and the corresponding factors.

Interestingly, in analogy to dropout, where each hidden unit is dropped independently with probability $1 - \theta$, in the

proposed randomized tensor decomposition, the columns of the factor matrices and the corresponding fibers of the core tensor are dropped independently and consequently the *rank* of the tensor decomposition is stochastically dropped.

Bernoulli CP randomized tensor regression An interesting special case of 5 is when the weight tensor $\tilde{\mathcal{W}}$ of the TRL is expressed using a CP decomposition. In that case, we set $\mathbf{M} = \mathbf{M}^{(0)} = \dots = \mathbf{M}^{(N)} = \text{diag}(\boldsymbol{\lambda})$, with, for any $k \in [0 \dots R]$, $\lambda_k \sim \text{Bernoulli}(\theta)$.

Then a randomized CP approximation is expressed as:

$$\tilde{\mathcal{W}} = \sum_{k=0}^{R-1} \tilde{\mathbf{U}}_k^{(0)} \circ \dots \circ \tilde{\mathbf{U}}_k^{(N)} \quad (8)$$

The above randomized CP decomposition on the weights is equivalent to the following formulation:

$$\begin{aligned} \tilde{\mathcal{W}} &= \sum_{k=0}^{R-1} \lambda_k \mathbf{U}_k^{(0)} \circ \dots \circ \mathbf{U}_k^{(N)} \\ &= \llbracket \boldsymbol{\lambda}; \mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N)} \rrbracket \end{aligned} \quad (9)$$

This is easy to see by looking at the individual elements of the sketched factors. Let $k \in [0 \dots N]$ and $i_k \in [0 \dots I_k], r \in [0 \dots R-1]$. Then $\tilde{\mathbf{U}}_{i_k, r}^{(k)} = \sum_{j=0}^{R-1} \mathbf{U}_{i_k, j}^{(k)} \mathbf{M}_{j, r}$. Since $\mathbf{M} = \text{diag}(\boldsymbol{\lambda})$, i.e. $\forall i, j \in [0 \dots R-1], \mathbf{M}_{ij} = 0$ if $i \neq j$, and λ_i otherwise, we get $\tilde{\mathbf{U}}_{i_k, r}^{(k)} = \lambda_r \mathbf{U}_{i_k, r}^{(k)}$. It follows that $\tilde{\mathcal{W}}_{i_0, i_1, \dots, i_N} = \sum_{r=0}^{R-1} \lambda_r \mathbf{U}_{i_0, r}^{(0)} \circ \dots \circ \lambda_r \mathbf{U}_{i_N, r}^{(N)}$. Since $\lambda_r \in \{0, 1\}$, we have $\tilde{\mathcal{W}}_{i_0, i_1, \dots, i_N} = \sum_{r=0}^{R-1} \lambda_r (\mathbf{U}_{i_0, r}^{(0)} \circ \dots \circ \mathbf{U}_{i_N, r}^{(N)})$.

Based on the previous stochastic regularization, for an activation tensor \mathcal{X} and a corresponding label vector \mathbf{y} , the optimization problem for our tensor regression layer with stochastic regularization is given by:

$$\min_{\mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N)}} \|\mathbf{y} - \frac{1}{\theta} \langle \llbracket \boldsymbol{\lambda}; \mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N)} \rrbracket, \mathcal{X} \rangle_{N-1}\|_F^2 \quad (10)$$

In addition, the above stochastic optimization problem can be rewritten as a deterministic regularized problem:

$$\begin{aligned} \mathbb{E}_{\boldsymbol{\lambda}} \left[\min_{\mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N)}} \|\mathbf{y} - \frac{1}{\theta} \langle \llbracket \boldsymbol{\lambda}; \mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N)} \rrbracket, \mathcal{X} \rangle_{N-1}\|_F^2 \right] \\ = \min_{\mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N)}} \|\mathbf{y} - \langle \llbracket \mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N)} \rrbracket, \mathcal{X} \rangle_{N-1}\|_F^2 \\ + \left(\frac{1-\theta}{\theta} \right) \sum_{k=0}^{R-1} \left(\prod_{i=0}^N \|\mathbf{U}_{:,k}^{(i)}\|_2^2 \right) \end{aligned} \quad (11)$$

This is easy to see by considering the equivalent rewriting of the above optimization problem, using the mode- N unfolding of the weight tensor. Equation 10 then becomes:

$$\min_{\mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N)}} \|\mathbf{y} - \frac{1}{\theta} \mathbf{U}^{(N)} \text{diag}(\boldsymbol{\lambda}) (\mathbf{U}^{(-N)})^\top \text{vec}(\mathcal{X})\|_F^2$$

with $\mathbf{U}^{(-N)} = (\mathbf{U}^{(0)} \circ \dots \circ \mathbf{U}^{(N)})$. The result can then be obtained following [31, Lemma A.1].

5. Experimental evaluation

In this section, we introduce the experimental setting, databases used, and implementation details. We experimented on several datasets, across various tasks, namely image classification and MRI-based regression. All methods were implemented * using PyTorch [33] and TensorLy [24].

5.1. Numerical experiments on synthetic data

Here, we empirically demonstrate the equivalence between our stochastic rank regularization and the deterministic regularization based formulation of the dropout.

To do so, we first created a random regression weight tensor \mathcal{W} to be a third order tensor of size $(25 \times 25 \times 25)$, formed as a low-rank Kruskal tensor with 15 components, the factors of which were sampled from an i.i.d. Gaussian distribution. We then generated a tensor of 10000 random samples, \mathcal{X} of size $(10000 \times 25 \times 25 \times 25)$, the elements of which were sampled from a Normal distribution. Finally, we constructed the corresponding response array \mathbf{y} of size 10000 as: $\forall i \in [1 \dots 1500], \mathbf{y}_i = \langle \mathcal{X}_i, \mathcal{W} \rangle$. Using the same regression weight tensor and same procedure, we also generated 1000 testing samples and labels.

We use this data to train a rank-15 CP R-TRL, with both our Bernoulli stochastic formulation (equation 10) and its deterministic counter-part (equation 11). We train for 500 epochs, with a batch-size of 200, and an initial learning rate of $10e-4$, which we decrease by a factor of 10 every 200 epochs. Figure 2 shows the loss function as a function of the epoch number. As expected, both formulations are identical.

5.2. Results on image classification

In the image classification setting, we perform a thorough study of our method on the CIFAR100 dataset. In particular, we empirically compare our approach to both standard baseline and traditional tensor regression, and assess the robustness of each method in the face of adversarial noise. We also report results for large-scale image classification on the ImageNet dataset.

CIFAR-100 [25] consists of 60,000 32×32 RGB images in 100 classes, divided into 50,000 images for training and 10,000 for testing. We pre-processed the data by centering and scaling each image and then augmented the training images with random cropping and random horizontal flipping.

We compare the randomized tensor regression layer to full-rank tensor regression, average pooling and a fully-connected layer in an 18-layer residual network (ResNet) [16]. For all networks, we used a batch size of 128 and

*code will be released upon acceptance to reproduce all results

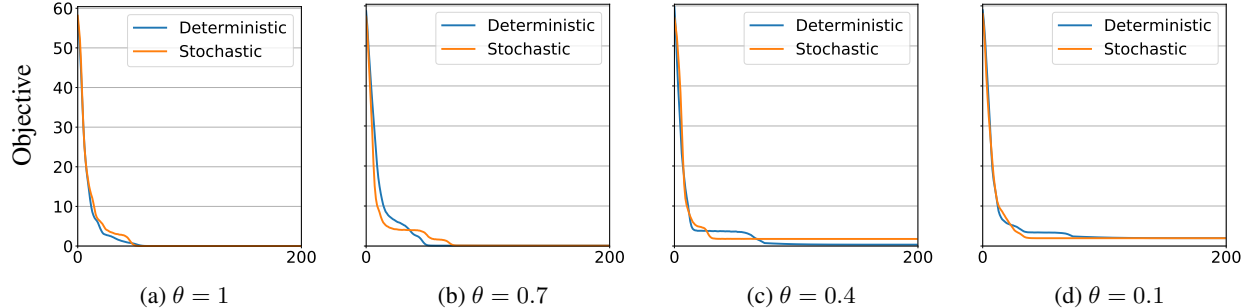


Figure 2: **Experiment on synthetic data:** loss of the R-TRL as a function of the number of epochs for the stochastic version (orange) and the deterministic one based on the regularized objective function (blue). As expected, both formulations are empirically the same.

trained for 400 epochs, and minimized the cross-entropy loss using stochastic gradient descent (SGD). The initial learning rate was set to 0.01 and lowered by a factor of 10 at epochs 150, 250 and 350. We used a weight decay (L_2 penalty) of 10^{-4} and a momentum of 0.9.

ImageNet [11] is a large-scale dataset for image classification composed of 1.2 million training images and 50,000 images for validation. We evaluate the classification error in terms of top-1 and top-5 classification accuracy on a 224×224 single center crop from the raw input images. For training, we use a ResNet101 and follow the same procedure and setting as [16, 23].

Results Table 1 presents results obtained on the CIFAR-100 dataset, on which our method matches or outperforms other methods, including the same architectures without R-TRL. Our regularization method makes the network more robust by reducing over-fitting, thus allowing for superior performance on the testing set.

ResNet classification	Accuracy
FC	75.88 %
FC + dropout	75.84 %
Tucker	76.02 %
CP	75.77 %
Randomized Tucker	76.05 %
Randomized CP	76.19 %

Table 1: **Classification accuracy for CIFAR-100** with a ResNet and various regression layers for classification.

A natural question is whether the model is sensitive to the choice of rank and θ (or drop rate when sampling with repetition). To assess this, we show the performance as a function of both rank and θ in figure 3. As can be observed, there is a large surface for which performance remains the same while decreasing both parameters (note the logarithmic scale for the rank). This means that, in practice, choosing good values for these is not a problem.

We also test method on the more challenging ImageNet dataset. The results, Table 2 show that our method outperforms the baselines, with higher classification accuracy even for large values of θ .

Robustness to adversarial attacks: We test for robustness to adversarial examples produced using the Fast Gradient Sign Method [29] in Foolbox [35]. In this method, the sign of the optimization gradient multiplied by the perturbation magnitude is added to the image in a single iteration. The perturbations we used are of magnitudes $\lambda \times 10^{-3}$, $\lambda \in \{1, 2, 4, 8, 16, 32, 64, 128\}$.

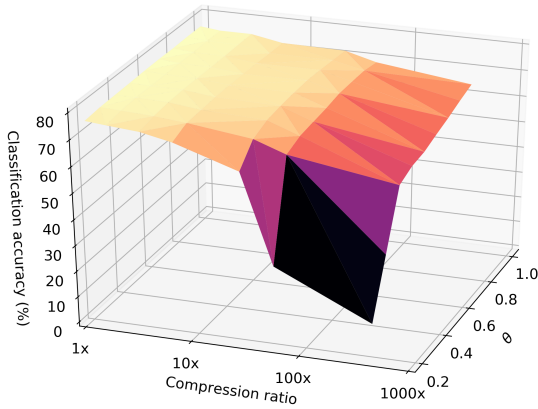
In addition to improving performance by reducing over-fitting, our proposed stochastic regularization makes the model more robust to perturbations in the input, for both random noise and adversarial attacks.

We tested the robustness of our models to adversarial attacks, when trained in the same configuration. In figure 4, we report the classification accuracy on the test set, as a function of the added adversarial noise. Specifically, we sample 1,000 images from the test set [3]. The models were trained *without* any adversarial training, on the training set, and adversarial noise was added to the test samples using the Fast Gradient Sign method. Our model is much more robust to adversarial attacks. Finally, we perform a thorough comparison of the various regularization strategies, the results of which can be seen in figure 5.

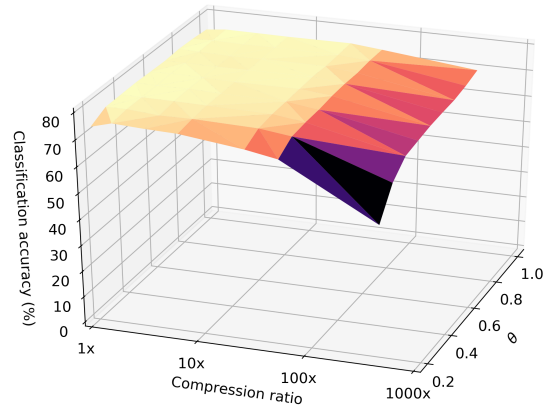
We perform a similar experiment on ImageNet (Figure 6), using Tucker R-TRL with half the full-rank and arrive at similar conclusion.

5.3. Phenotypic trait prediction from MRI data

In the regression setting, we investigate the performance of our R-TRL in a challenging, real-life application on a very large-scale dataset. This case is particularly interesting since MRI volumes are large 3D tensors, all modes of which carry important information. The spatial information is traditionally discarded during the flattening process, which we avoid by using a tensor regression layer.



(a) Bernoulli Tucker R-TRL



(b) Replacement Tucker R-TRL

Figure 3: **CIFAR-100 test accuracy** as a function of the compression ratio (logarithmic scale) and the Bernoulli probability θ (left) or the drop rate (right). There is a large region for which dropping both the rank and θ does not hurt performance.

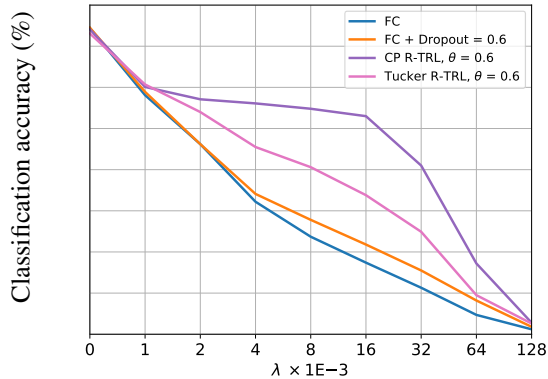


Figure 4: **Robustness to adversarial attacks on CIFAR100** using Fast Gradient Sign attacks of various models. Our randomized TRL architecture is much more robust to adversarial attacks, even though adversarial training was not used.

Model	θ	Accuracy	
		Top-1 (%)	Top-5 (%)
ResNet	\times	77.1	93.4
Ours	0.9	77.7	93.7
Ours	0.8	77.7	93.7
Ours	0.7	77.4	93.6
Ours	0.6	78.0	93.8

Table 2: **Classification accuracy on ImageNet** with Bernoulli Tucker R-TRL.

The UK Biobank brain MRI dataset is the world’s largest MRI imaging database of its kind [39]. The aim of the UK Biobank Imaging Study is to capture MRI scans of vital organs for 100,000 primarily healthy individuals by

2022. Associations between these images and lifestyle factors and health outcomes, both of which are already available in the UK Biobank, will enable researchers to improve diagnoses and treatments for numerous diseases. The data we use here consists of T1-weighted $182 \times 218 \times 182$ MR images of the brain for 7,500 individuals captured on a 3 T Siemens Skyra system. 5,700 are used for training, 800 are used for validation and 1,000 samples are used to test. The target label is the age for each individual at the time of MRI capture. We use skull-stripped images that have been aligned to the MNI152 template [19] for head-size normalization. We then center and scale each image to zero mean and unit variance for intensity normalization.

Architecture	Regression	MAE
3D-ResNet	FC	3.23 years
3D-ResNet	Tucker	2.99 years
Ours	Randomized Tucker	2.77 years
Ours	Randomized CP	2.58 years

Table 3: **Classification accuracy for UK Biobank MRI**. The ResNet models with R-TRL performs significantly outperforms the version with a fully-connected (FC) layer.

Results: For MRI-based experiments we implement an 18-layer ResNet with three-dimensional convolutions. We minimize the mean squared error using Adam [21], starting with an initial learning rate of 10^{-4} , reduced by a factor of 10 at epochs 25, 50, and 75. We train for 100 epochs with a mini-batch size of 8 and a weight decay (L_2 penalty) of 5×10^{-4} . For Tucker-based R-TRL we used a tensor with rank $128 \times 6 \times 7 \times 6$. For CP-based R-TRL we used a Kruskal tensor with 82 components. As previously ob-

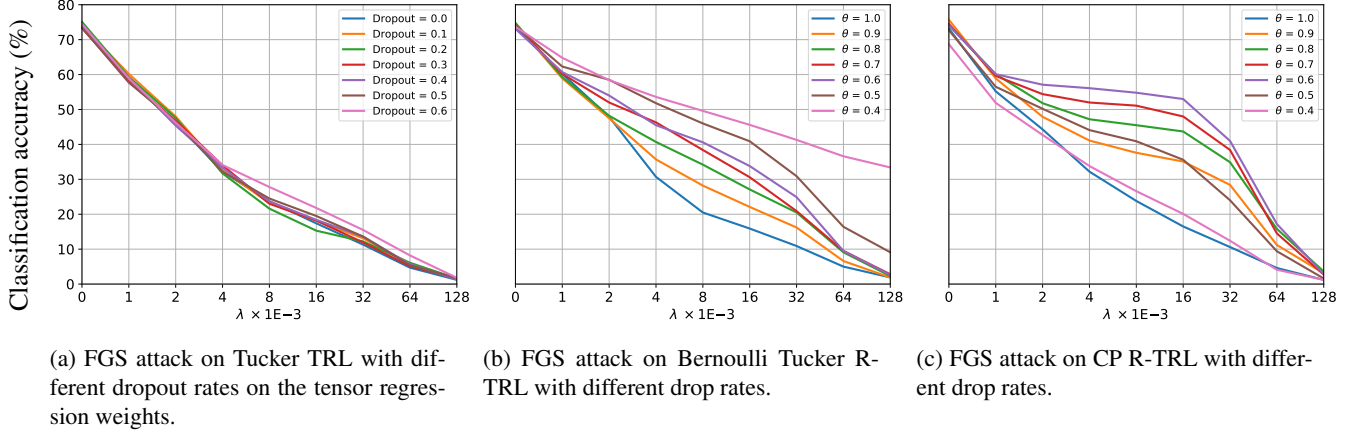


Figure 5: **Robustness to adversarial attacks**, measured by adding adversarial noise to the test images, using the Fast Gradient Sign, on CIFAR-100 and Bernoulli drop. We compare a Tucker tensor regression layer with dropout applied to the regression weight tensor 5a to our randomized TRL, both in the Tucker (Subfig. 5b) and CP (Subfig. 5c) case.

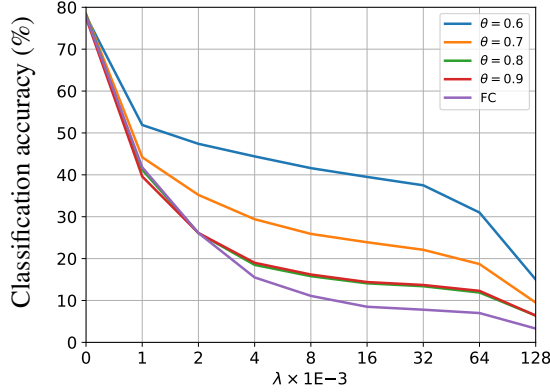


Figure 6: **Robustness to adversarial attacks on ImageNet**. Our R-TL architecture is much more robust to adversarial attacks, even though adversarial training was not used.

served, our randomized tensor regression network outperforms the ResNet baseline by a large margin, Table 3. To put this into context, the current state-of-art for convolutional neural networks on age prediction from brain MRI on most datasets is an MAE of around 3.6 years [8].

Robustness to noise: We tested the robustness of our model to white Gaussian noise added to the MRI data. Noise in MRI data typically follows a Rician distribution but can be approximated by a Gaussian for signal-to-noise ratios (SNR) greater than 2 [15]. As both the signal (MRI voxel intensities) and noise are zero-mean, we define $SNR = \frac{\sigma_{\text{signal}}^2}{\sigma_{\text{noise}}^2}$, where σ is the variance. We incrementally increase the added noise in the test set and compare the error rate of the models.

The ResNet with R-TL is significantly more robust to added white Gaussian noise compared to the same architec-

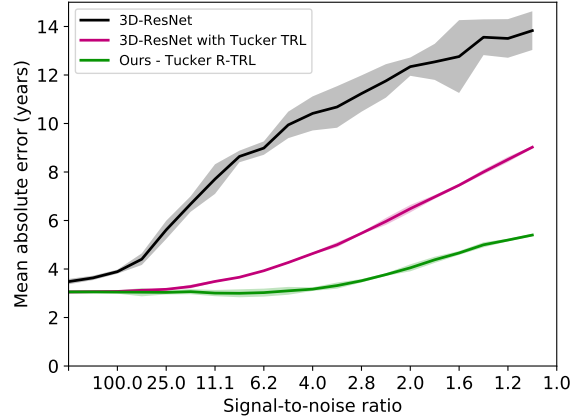


Figure 7: **Age prediction error on the MRI test set** as a function of increased added Gaussian noise. Shaded regions indicate 95% confidence intervals for 5 independent runs.

tures without it (figure 7). At signal-to-noise ratios below 10, the accuracy of a standard fully-connected ResNet is worse than a naive model that predicts the mean of training set (MAE = 7.9 years). Brain morphology is an important attribute that has been associated with various biological traits including cognitive function and overall health [34, 40]. By keeping the structure of the brain represented in MRI in every layer of the architecture, the model has more information to learn a more accurate representation of the entire input. Randomly dropping the rank forces the representation to be robust to confounds. This a particularly important property for MRI analysis since intensities and noise artifacts can vary significantly between MRI scanners [45]. Randomized tensor regression layers enable both more accurate and more robust trait predictions from MRI that can consequently lead to more accurate disease diagnoses.

6. Conclusion

We introduced a novel randomized tensor decomposition, suitable for end-to-end training of tensor regression layer. By adding stochasticity on the rank during training, it renders the network more robust and lead to better performance. This results in networks that are more resilient to noise, both adversarial and random, without any adversarial training. Our results demonstrate superior performance on a variety of real-life, large scale challenging tasks, including MRI data and images, as well as much better robustness to noise. Finally, we also prove the link between this randomized TRL and dropout on the deterministic version.

Acknowledgements

This research has been conducted using the UK Biobank Resource under Application Number 18545.

References

- [1] C. Battaglini, G. Ballard, and T. G. Kolda. A practical randomized cp tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 39(2):876–901, 2018. 2
- [2] A. Bietti, G. Mialon, D. Chen, and J. Mairal. A kernel perspective for regularizing deep neural networks. 2019. 1
- [3] W. Brendel, J. Rauber, and M. Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017. 6
- [4] X. Cao, G. Rabusseau, and J. Pineau. Tensor regression networks with various low-rank tensor approximations. *CoRR*, abs/1712.09520, 2017. 3
- [5] R. Caruana, S. Lawrence, and C. L. Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems*, pages 402–408, 2001. 1
- [6] W. Chen, J. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing convolutional neural networks in the frequency domain. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1475–1484. ACM, 2016. 2
- [7] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, and S.-F. Chang. An exploration of parameter redundancy in deep networks with circulant projections. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2857–2865, 2015. 1, 2
- [8] J. H. Cole, R. P. Poudel, D. Tsagkrasoulis, M. W. Caan, C. Steves, T. D. Spector, and G. Montana. Predicting brain age with deep learning from raw imaging data results in a reliable and heritable biomarker. *NeuroImage*, 163:115–124, 2017. 8
- [9] J. H. Cole, S. J. Ritchie, M. E. Bastin, M. V. Hernández, S. M. Maniega, N. Royle, J. Corley, A. Pattie, S. E. Harris, Q. Zhang, et al. Brain age predicts mortality. *Molecular psychiatry*, 2017. 2
- [10] A. Daniely, N. Lazic, Y. Singer, and K. Talwar. Sketching and neural networks. *arXiv preprint arXiv:1604.05753*, 2016. 2
- [11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 6
- [12] N. B. Erichson, K. Manohar, S. L. Brunton, and J. N. Kutz. Randomized cp tensor decomposition. *arXiv preprint arXiv:1703.09074*, 2017. 2
- [13] K. Franke, E. Luders, A. May, M. Wilke, and C. Gaser. Brain maturation: predicting individual brainage in children and adolescents using structural mri. *Neuroimage*, 63(3):1305–1312, 2012. 2
- [14] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 1
- [15] H. Gudbjartsson and S. Patz. The rician distribution of noisy mri data. *Magnetic resonance in medicine*, 34(6):910–914, 1995. 8
- [16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 5, 6
- [17] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 1
- [18] D. Jakubovitz and R. Giryes. Improving dnn robustness to adversarial attacks using jacobian regularization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 514–529, 2018. 1
- [19] M. Jenkinson, P. Bannister, M. Brady, and S. Smith. Improved optimization for the robust and accurate linear registration and motion correction of brain images. *Neuroimage*, 17(2):825–841, 2002. 7
- [20] S. P. Kasiviswanathan, N. Narodytska, and H. Jin. Deep neural network approximation using tensor sketching. *arXiv preprint arXiv:1710.07850*, 2017. 2
- [21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7
- [22] J. Kossaifi, A. Khanna, Z. Lipton, T. Furlanello, and A. Anandkumar. Tensor contraction layers for parsimonious deep nets. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017 IEEE Conference on, pages 1940–1946. IEEE, 2017. 3
- [23] J. Kossaifi, Z. C. Lipton, A. Khanna, T. Furlanello, and A. Anandkumar. Tensor regression networks. *CoRR*, abs/1707.08308, 2018. 1, 2, 3, 6
- [24] J. Kossaifi, Y. Panagakis, and M. Pantic. Tensorly: Tensor learning in python. *arXiv preprint arXiv:1610.09555*, 2016. 5
- [25] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 5
- [26] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1
- [27] A. Krogh and J. A. Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992. 1

- [28] J. Kukačka, V. Golkov, and D. Cremers. Regularization for deep learning: A taxonomy. *arXiv preprint arXiv:1710.10686*, 2017. 2
- [29] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016. 6
- [30] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015. 1
- [31] P. Mianjy, R. Arora, and R. Vidal. On the implicit bias of dropout. In J. Dy and A. Krause, editors, *International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 3540–3548, Stockholm Sweden, 10–15 Jul 2018. PMLR. 5
- [32] S. J. Nowlan and G. E. Hinton. Simplifying neural networks by soft weight-sharing. *Neural computation*, 4(4):473–493, 1992. 1
- [33] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017. 5
- [34] A. Pfefferbaum, D. H. Mathalon, E. V. Sullivan, J. M. Rawles, R. B. Zipursky, and K. O. Lim. A quantitative magnetic resonance imaging study of changes in brain morphology from infancy to late adulthood. *Archives of neurology*, 51(9):874–887, 1994. 8
- [35] J. Rauber, W. Brendel, and M. Bethge. Foolbox: a python toolbox to benchmark the robustness of machine learning models (2017). URL <http://arxiv.org/abs/1707.04131>, 2017. 6
- [36] S. Scardapane, D. Comminiello, A. Hussain, and A. Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017. 1
- [37] N. D. Sidiropoulos, E. E. Papalexakis, and C. Faloutsos. Parallel randomly compressed cubes: A scalable distributed architecture for big tensor decomposition. *IEEE Signal Processing Magazine*, 31(5):57–70, 2014. 2
- [38] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 1, 2
- [39] C. Sudlow, J. Gallacher, N. Allen, V. Beral, P. Burton, J. Danesh, P. Downey, P. Elliott, J. Green, M. Landray, et al. Uk biobank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age. *PLoS medicine*, 12(3):e1001779, 2015. 7
- [40] G. E. Swan, C. DeCarli, B. Miller, T. Reed, P. Wolf, L. Jack, and D. Carmelli. Association of midlife blood pressure to late-life cognitive decline and brain morphology. *Neurology*, 51(4):986–993, 1998. 8
- [41] C. Tai, T. Xiao, Y. Zhang, X. Wang, et al. Convolutional neural networks with low-rank regularization. *arXiv preprint arXiv:1511.06067*, 2015. 1, 2
- [42] C. E. Tsourakakis. Mach: Fast randomized tensor decompositions. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 689–700. SIAM, 2010. 2
- [43] N. Vervliet, O. Debals, L. Sorber, and L. De Lathauwer. Breaking the curse of dimensionality using decompositions of incomplete tensors: Tensor-based scientific computing in big data analysis. *IEEE Signal Processing Magazine*, 31(5):71–79, 2014. 2
- [44] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066, 2013. 1
- [45] L. Wang, H.-M. Lai, G. J. Barker, D. H. Miller, and P. S. Tofts. Correction for variations in mri scanner sensitivity in brain studies with histogram matching. *Magnetic resonance in medicine*, 39(2):322–327, 1998. 8
- [46] Y. Wang, H.-Y. Tung, A. J. Smola, and A. Anandkumar. Fast and guaranteed tensor decomposition via sketching. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 991–999. 2015. 2
- [47] X. Yu, T. Liu, X. Wang, and D. Tao. On compressing deep models by low rank and sparse decomposition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 67–76. IEEE, 2017. 1, 2
- [48] L. Yuan, C. Li, J. Cao, and Q. Zhao. Randomized tensor ring decomposition and its application to large-scale data reconstruction. *arXiv preprint arXiv:1901.01652*, 2019. 2
- [49] Y. Zhang, J. D. Lee, and M. I. Jordan. l1-regularized neural networks are improperly learnable in polynomial time. In *International Conference on Machine Learning*, pages 993–1001, 2016. 1
- [50] G. Zhou, A. Cichocki, and S. Xie. Decomposition of big tensors with low multilinear rank. *arXiv preprint arXiv:1412.1885*, 2014. 2